逻辑与神经网络之间的桥

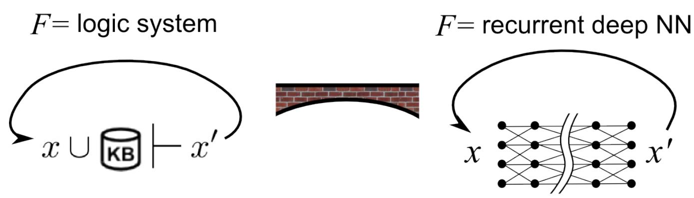
甄景贤 (King-Yin Yan)

General.Intelligence@Gmail.com

有个问题希望数学专业的人帮手解决一下：

人工智能基本上分为两大阵营：**逻辑 AI** (logic-based AI) 和 **神经网络** (connectionism)。

逻辑 AI 那边，「结构」很精细，但学习算法太慢；我的目的是建立一道「桥」，将逻辑 AI 的某部分结构**转移**到神经网络那边，这样就可以融合两边的好处。



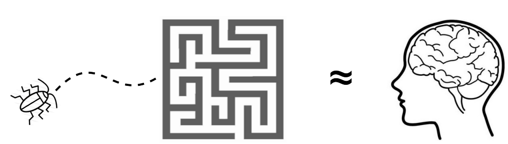
(1)

这个问题搞了很久都未能解决，因为逻辑 AI 那边的结构不是一般常见的数学结构，单是要要表述出来也有很大困难。这表述个问题大概可以叫做 algebraization of logic，现在仍没有公认的答案，但有不少「部分成功」的理论（下述）。

我首先解释 connectionism 那边的结构，然后再解释 logic 那边的结构。

1 Connectionist architecture

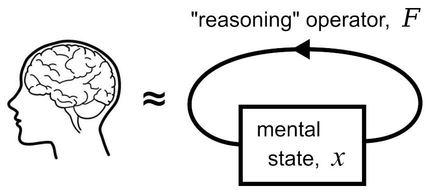
用一个比喻来说，就是用强化学习去控制一隻智慧生物，在「思考空间」的迷宫中找出路：



(2)

强化学习特别适合解决这类问题，可以参看我写的 tutorial。

关键是将「思考」看成是一个动态系统 (dynamical system)，它运行在思维状态 (mental states) 的空间中：



(3)

举例来说，一个思维状态可以是以下的一束命题：

- 我在我的房间内，正在写一篇论文。
- 我正在写一句句子的开头：「我在我的房间内，....」
- 我将会写一个动词词组 (verb phrase)：「正在写....」

思考的过程就是从一个思维状态 过渡 (transition) 到另一个思维状态。就算我现在说话，我的脑子也是靠思维状态记住我说话说到句子结构的哪部分，所以我才能组织句子的语法。

思维状态是一支向量 $\boldsymbol{x} \in X$，X 是全体思维空间，思考算子 (reasoning operator) $F : X \to X$ 是一个 endomorphism。

一个**动态系统 (dynamical system)** 可以用以下方法定义：

$$\text{离散时间：} \qquad \boldsymbol{x}_{n+1} = \boldsymbol{F}(\boldsymbol{x}_n) \tag{4}$$

$$\text{连续时间：} \qquad \dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) \tag{5}$$

其中 f 也可以随时间改变。如果 f 不依赖时间，则系统是 time-invariant （定常的），形式上如 (**??**) 那种微分方程叫作 autonomous （自主的）。为方便起见，有时我会滥用 F 和 f 的表述（不区分连续和离散）。

一个（连续时间的）**控制系统 (control system)** 定义为：

$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \tag{6}$$

其中 $\boldsymbol{u}(t)$ 是控制向量。控制论的目的就是找出最好的 $\boldsymbol{u}^*(t)$，令系统由初始状态 $\boldsymbol{x}_0$ 去到终点状态 $\boldsymbol{x}_\perp$。

动力系统 ?? 有标准的 Hamiltonian 力学解释，这可以看成是「思维动力学」。系统的**相位空间**有 symplectic 结构，可以利用这结构更有效地解微分方程 **??**。

而**控制系统 ??** 中，每次到达新状态 x，外部会给出一个奖励 $r(x)$，它对应於 Lagrangian，其单位是**能量**。（换句话说，智能系统的欲望是用能量量度的；详见 tutorial。）

动态规划 (dynamic programming) 的中心思想是

注意：人工智能中的 **A* search**，是动态规划的一个特例。换句话说，用动态规划在某个空间中「漫游」，可以模拟 best-first 搜寻的功能。

我们的目标是学习 $F \in \{$ 无限维的算子空间 $\}$。实践上 F 可以用 deep learning network 代表，换句话说 F 就是一个有很多 parameters 的非线性算子 (= 神经网络)。

一个**神经网络**基本上是：

$$F(\boldsymbol{x}) = \sigma(W_1 \sigma(W_2 ... \sigma(W_L\ \boldsymbol{x}))) \tag{7}$$

其中 L 是层数，W 是每层的权重**矩阵**，σ 是对每个分量的 sigmoid function（其作用是赋予非线性）。

所有 W 的分量存在於一个**高维流形**之中，**学习**的目的就是在这流形中寻找最好的 W^*。在这流形上可以定义一个**度量** (metric)（例如引入随机性之后，用 Kullback–Leibler divergence 定义），这是 information geometry 的做法，用 geodesic 可以令学习快点。

在这框架下，智能系统的运作可以分开成两方面：思考 和 学习。

思考即是根据已学得的知识（知识储存在 RNN 里），在思维空间中找寻 $\boldsymbol{x}$ 最优的轨迹，方法是用控制论计算 $\boldsymbol{u}^*$。$\boldsymbol{x}$ 的轨迹受 RNN 约束（系统只能依据「正确」的知识去思考），但思考时 RNN 是不变的。

学习就是学习神经网络 RNN 的 weights W。此时令 $u = 0$，即忽略控制论方面。

而很明显，「自由」的 F 算子没有「内部结构」，它能够学习的就像是甲甴那样的、简单的「条件反射」行为。如果要达到人类的智慧，则要学习很久（到时我们都死了）。

所以问题就是要赋予 F 更多的**结构**，特别是逻辑结构。直观地说，越多的结构令**搜寻空间**越小，学习会越快。这是机器学习里面 inductive bias 的标准做法。

2 Logic-based AI

用数理逻辑 (mathematical logic) 模拟人的思想是可行的，例如有 deduction, abduction, induction 等这些模式，详细可见《Computational logic and human thinking》by Robert Kowalski, 2011. 这些方面不影响本文的阅读。值得一提的是，作者 Kowalski 是 logic programming，特别是 Prolog，的理论奠基人之一。

在经典逻辑 AI 中，「思考」是透过一些类似以下的步骤：

$$\text{前提} \vdash \text{结论} \tag{8}$$

$$\boxed{\text{今天早上下雨}} \vdash \boxed{\text{草地是湿的}} \tag{9}$$

亦即由一些命题(propositions) 推导到另一些命题。

推导必须依靠一些逻辑的法则命题 (rule propositions)，所谓「法则」是指命题里面带有 x 这样的变量(variables)：

$$\boxed{\text{地方 x 下雨}} \wedge \boxed{\text{x 是露天的}} \vdash \boxed{\text{地方 x 是湿的}} \tag{10}$$

这些法则好比「逻辑引擎」的燃料，没有燃料引擎是不能推动的。

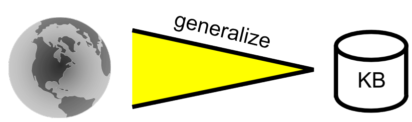
注意：命题里面的 x，好比是有「洞」的命题，它可以透过 substitution 代入一些实物 (objects)，而变成完整的命题。这种「句子内部」(sub-propositional) 的结构可以用 predicate logic （谓词逻辑）表达，但暂时不需要理会这些细节。

「所有人失恋了都会不开心」：

$$\forall z.\ \not\heartsuit(z) \rightarrow ☹(z) \tag{11}$$

在数理逻辑中这是一条 公理 (axiom)，但在 AI 中这些公理是从主体的经验中**学习**出来的，我们仍沿用「公理」这术语。在 AI 术语中，公理的集合叫 knowledge base，记作 KB。注意 KB 是一堆 **formulas** 的集合。

Logic-based AI 可以看成是将世界的「模型」压缩成一个 KB：



(12)

世界模型是由大量的逻辑式子经过组合而生成的，有点像向量空间是由其「基底」生成；但这生成过程在逻辑中特别复杂，所以符号逻辑具有很高的压缩比，但要学习一套逻辑 KB，则相应地也有极高的复杂度。

$$x \cup \text{KB} \vdash x' \tag{13}$$

逻辑 AI 的结构有两部分：

- 由一些命题推导出另一些命题
- 逻辑变量的处理

以下分别介绍。

2.1 由一些命题推导出另一些命题

命题也有内部结构（即命题可以由概念原子组合而成），但我们先从最简单情况谈起，即**命题逻辑**。

最简单的经典命题逻辑，是 Boolean propositional logic，它的**代数形式**是我们熟悉的 Boolean algebra，二者几乎没有分别（纯粹逻辑符号和代数符号的对应）。

在 Boolean algebra 可以定义一种 ideal I：

- If $a, b \in I$ then $a \wedge b \in I$
- If $a \in I$ and $a \leq b$ then $b \in I$

其中 $a \leq b$ 表示 $a \Rightarrow b$（a 蕴涵 b）。

由上面可以看出，这个 ideal 其实是由某些元素 (命题) 生成的 逻辑后果(logical consequence)；换句话说，给定一个命题集 Γ，问 $\Gamma \overset{?}{\vdash} a$（从 Γ 可以推导出 a 吗?）就等於问 a 是不是 Γ 生成的 ideal membership 问题。也可以说，代数 ideal $\equiv$ 逻辑 consequence。（严格来说，consequence 对应的是 filter 的概念，而 filter 是 ideal 的 dual，因为 0 和 1 对应的倒错，但这不是重点。）

逻辑后果可以记作 $\vdash$ 或 Cn，Tarski 定义了 $\vdash$ （很明显）的特性：

- (reflexivity): $A \vdash A$ for every formula A
- (monotonicity): $A \vdash Q$ implies $A, B \vdash Q$
- ('cut'): $A \vdash B$ and $A, B \vdash Q$ implies $A \vdash Q$

以上是 Boolean logic 的代数化，但如果考虑 probabilistic logic 就更为复杂，需要用到 Bayesian networks，而 filter $\equiv$ consequence 的原理似乎不再适用。

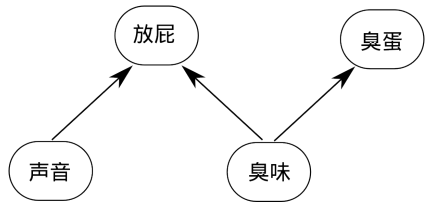
Bayesian network 的细节很麻烦，可以花整个研究生课程来讲。

重点是：Bayesian network 是由一些条件概率 (conditional probability) 的关系生成的。每个**节点**是一个命题，每个**连结**是一个条件概率关系，例如：

$$P(A|B, C, D, ...) = \vec{p} \tag{14}$$

其中 $\vec{p}$ 是一个 conditional probability table (CPT)。

对不起，用一个较粗俗的例子说明（在我多年的教学经验里，这是最易懂的例子）：



(15)

这个 Bayesian network 是由两个 CPT 生成的：

$$
\begin{aligned}
P(\text{放屁}|\text{臭味}, \text{声音}) &= \vec{p}_1 \\
P(\text{臭蛋}|\text{臭味}) &= \vec{p}_2
\end{aligned}
\tag{16}
$$

如果有「声音」又有「臭味」，则「有人放屁」的机率很高，而「臭蛋」的机率却会减少。换句话说，「臭蛋」的机率被扯到「放屁」那边去了；这个现象叫 “explaining away”，它说明 Bayesian network 中，所有节点都是 globally 相关的。所以，当求解 Bayesian network 的某个节点时，它的概率会是一连串很复杂的 sum-product 形式。看来用 Bayesian network 表示 $\vdash$ 的方法太复杂了。

可幸的是，可以用 Monte Carlo 方法求解 Bayesian network：开始时随机地指定节点的概率，然后随机地选取某些节点来作「**局部**」的 update；当随机 update 的次数趋近无限，节点的机率会收敛到正确的值。换句话说：这是一个 *local* 的计算 Bayesian network 的方法。

Knowledge-based model construction (KBMC) 这个术语较少人知道，但其实是最关键的结构；换句话说，就是从 KB 中抽出一组命题 Γ，去**组合**一个 model 或 proof tree，而这个 proof tree 的某个节点，就是新的结论。亦即 $\Gamma \vdash Q$。KBMC 的概念适用於经典逻辑也适用於 Bayesian networks。

2.2 逻辑变量的处理

这可能是最辣手的部分。

Alfred Tarski 提出了 cylindrical algebra 来解决「一阶谓词逻辑」的变量问题，但据说 Tarski 自己也觉得 cylindrical algebra 「不好用」。我觉得它比较难明，从略。

个人觉得比较好的做法是 Paul Halmos 的 algebraic logic。其中最关键的建构是：

$$
\begin{aligned}
\text{谓词} : \text{物体} &\rightarrow \text{命题空间} \\
\not\heartsuit : \not\heartsuit(\text{john}) &\mapsto \text{「阿 John 失恋」} \\
\not\heartsuit : \not\heartsuit(\text{pete}) &\mapsto \text{「阿 Pete 失恋」}
\end{aligned}
\tag{17}
$$

换句话说，predicate（谓词）就是一些将 object （逻辑中的物体，或「常项」，constants）映射到个别命题的**函数**。这些谓词函数可以有一个或多个**参数**，分别叫 monadic 和 polyadic predicate calculus。

最新的逻辑代数化方法来自範畴论，William Lawvere 在 1960's 年代提出（也就是 *Conceptual Mathematics* 这本书的作者之一）。他发现了 $\forall$ 和 $\exists$ 是 adjoint functors 的关系；这个 adjunction 比较难懂，有兴趣可以看 [1]。

References

1. Lawvere and Rosebrugh. *Sets for mathematics.* Cambridge, 2003.
2. Lo. Dynamical system identification by recurrent multilayer perceptrons. *Proceedings of the 1993 World Congress on Neural Networks*, 1993.
3. Siegelmann and Sontag. Turing computability with neural nets. *Applied Mathematics Letters, vol 4, p77-80*, 1991.